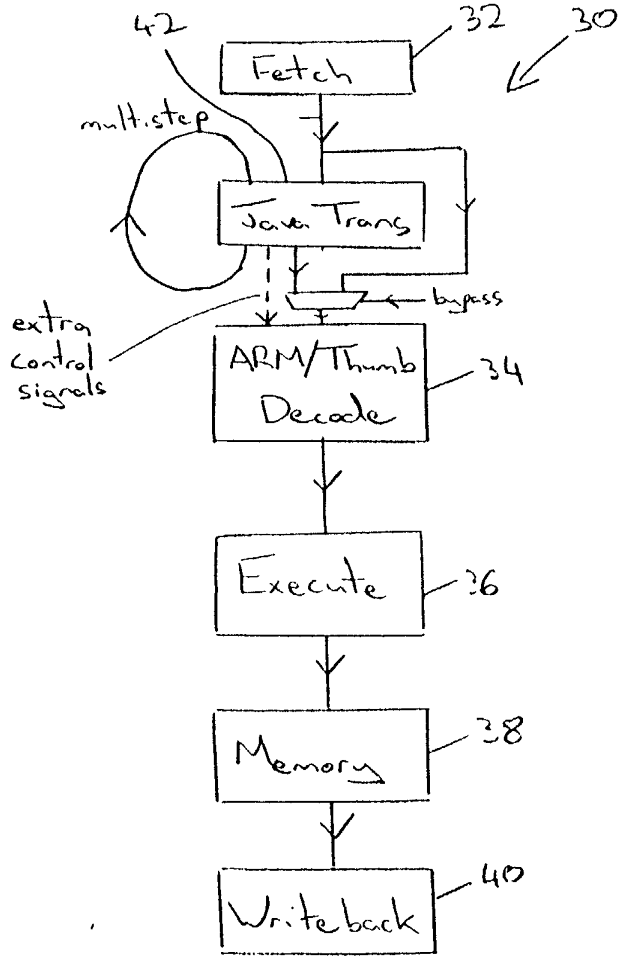
30
Fetch
32
42
multistep
Java Trans
bypass
extra control signals
ARM/Thumb Decode
34
Execute
36
Memory
38
Writeback
40

Fig. 1

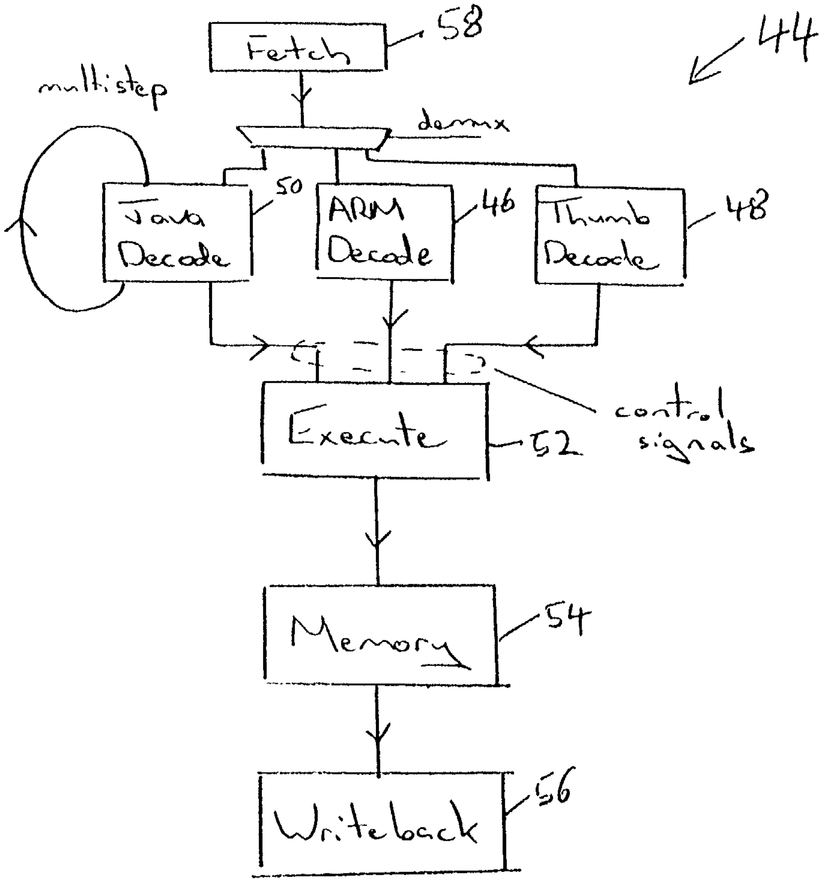
44
Fetch
58
multistep
demux
Java Decode
50
ARM Decode
46
Thumb Decode
48
control signals
Execute
52
Memory
54
Writeback
56

Fig. 2

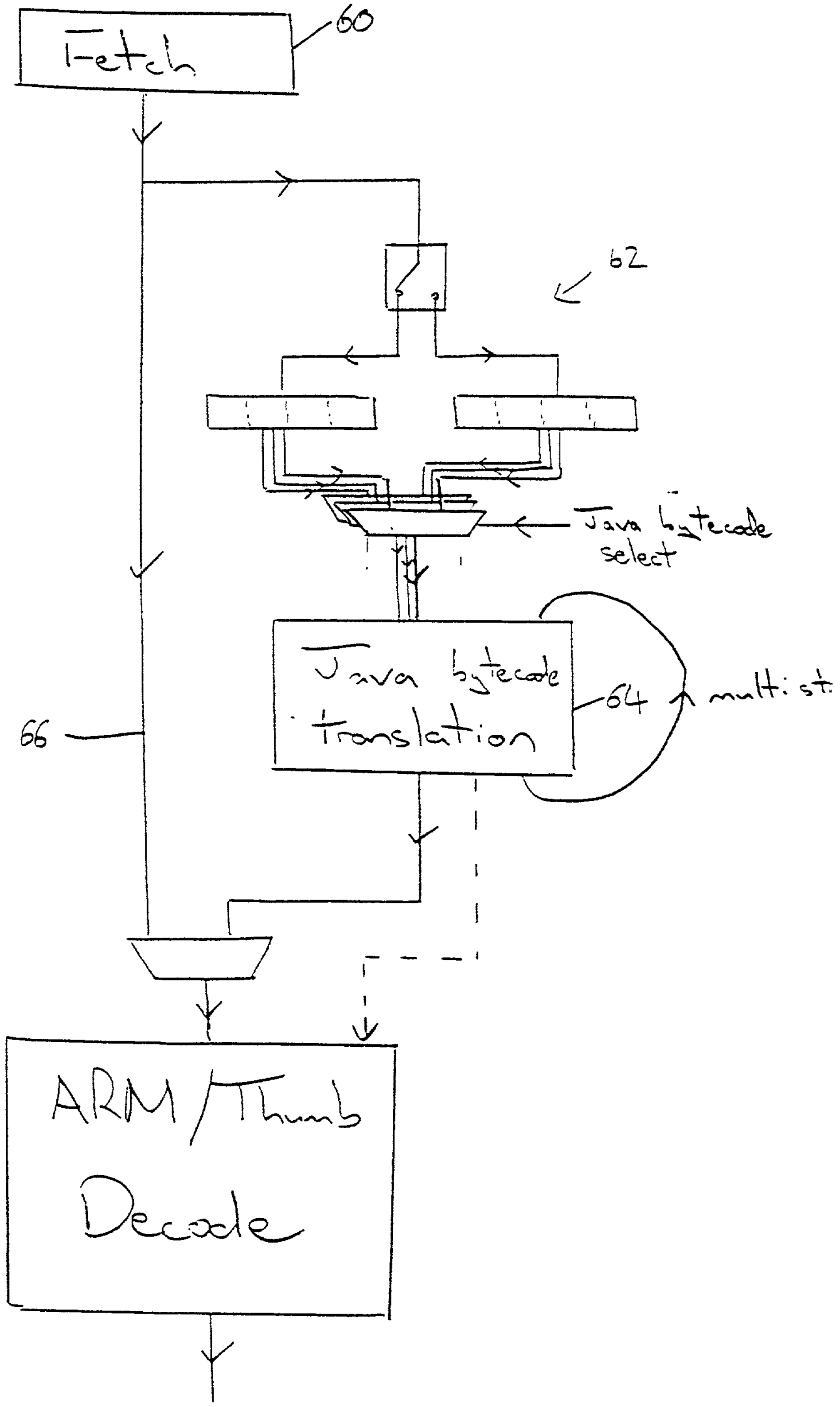
Fetch
60
62
Java bytecode select
Java bytecode translation
64
multi st.
66
ARM/Thumb Decode

Fig. 3

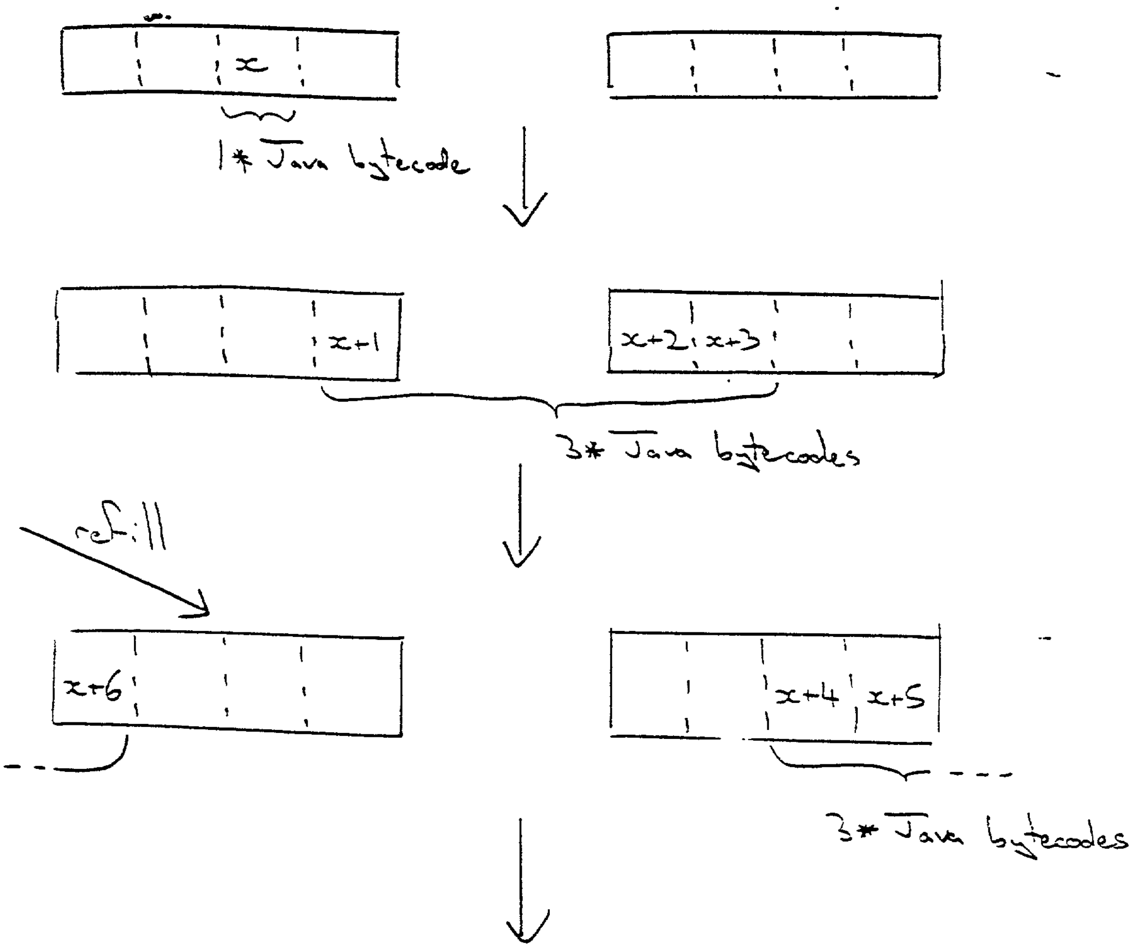
x
1* Java bytecode
x+1
x+2
x+3
3* Java bytecodes
refill
x+6
x+4
x+5
3* Java bytecodes

Fig. 4

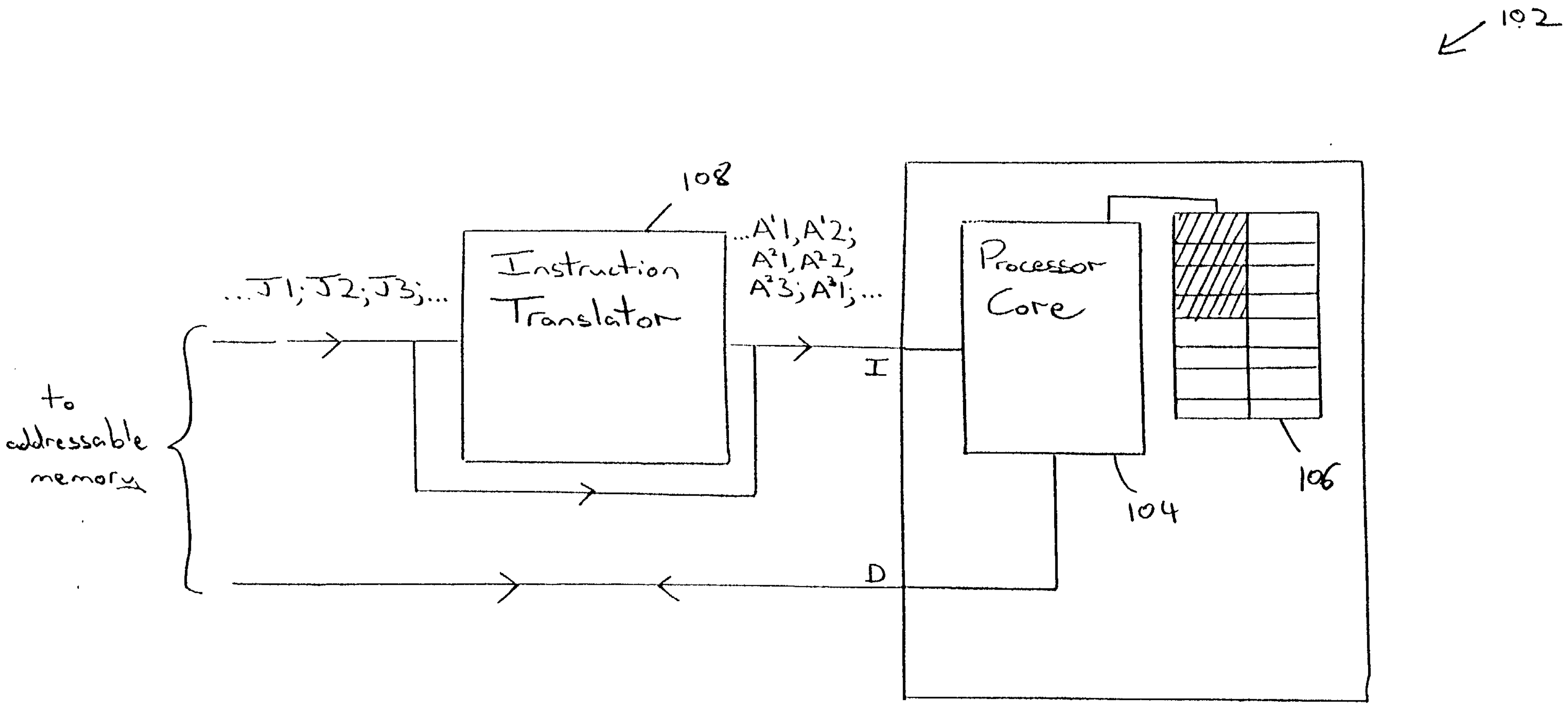
102
108
Instruction Translator
...J1; J2; J3;...
...A'1, A'2;
A²1, A²2,
A²3; A³1;...
I
D
Processor Core
104
106
to addressable memory

Fig. 5

Fig. 6

Java Instruction		iadd (RF=2, RF>∅)		iadd (RF=2, RF>1)		iadd (SA=-1)	
ARM Instruction(s)		LDR R∅[Rstack,#4]! (POP)		LDR R3 [Rstack,#4]! (POP)		ADD R3,R3,R∅	
State	00000		00100		01000		00111
R∅	E		SOA TOS		SOA TOS		E
R1	E		E		E		E
R2	E		E		E		E
R3	E		E		SOB TOS-1		(SOA+SOB) TOS

Java Instruction		lload¹ (RF=∅,RE=2)		lload² (RF=∅,RE>2)		lload² (RF=0,RE=2)	
ARM Instructions		LDR R1,[Rvars,#4] LDR R∅,[Rvars,#0]		STR R3 [Rstack],#4 (PUSH)		LDR R3,[Rvars,#4] LDR R2,[Rvars,#0]	
State	00111		01101		01001		10011
R∅	E		SOC TOS-1		SOC TOS-1		SOC TOS-3
R1	E		SOD TOS		SOD TOS		SOD TOS-2
R2	E		E		E		SOE TOS-1
R3	(SOA+SOB) TOS		(SOA+SOB) TOS-2		E		SOF TOS

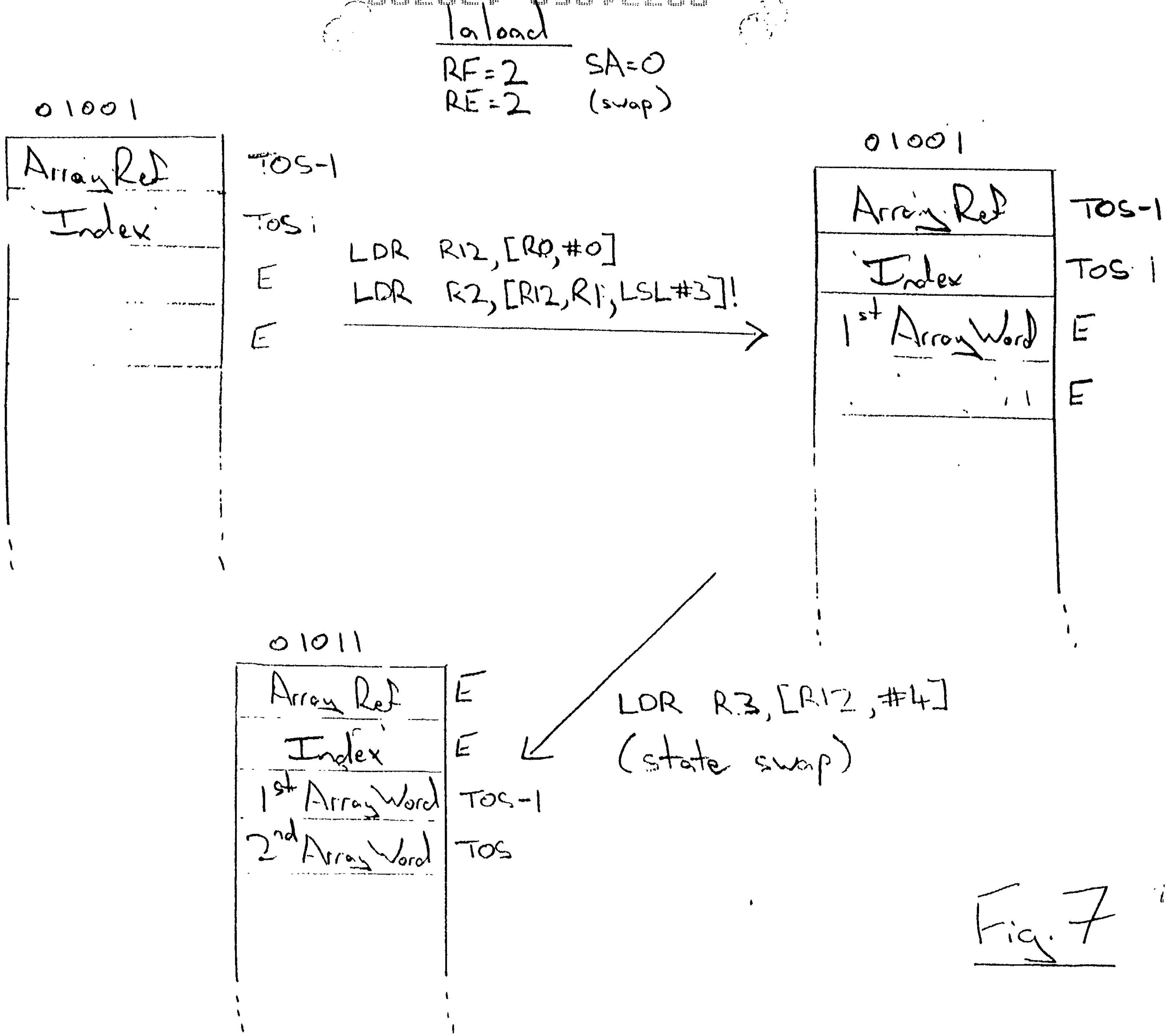
Iaload
RF=2
RE=2
SA=0
(swap)
01001
Array Ref
Index
TOS-1
TOS
E
E
LDR R12, [R0, #0]
LDR R2, [R12, R1, LSL#3]!
01001
Array Ref
Index
1st Array Word
TOS-1
TOS
E
E
LDR R3, [R12, #4]
(state swap)
01011
Array Ref
Index
1st Array Word
2nd Array Word
E
E
TOS-1
TOS

Fig. 7

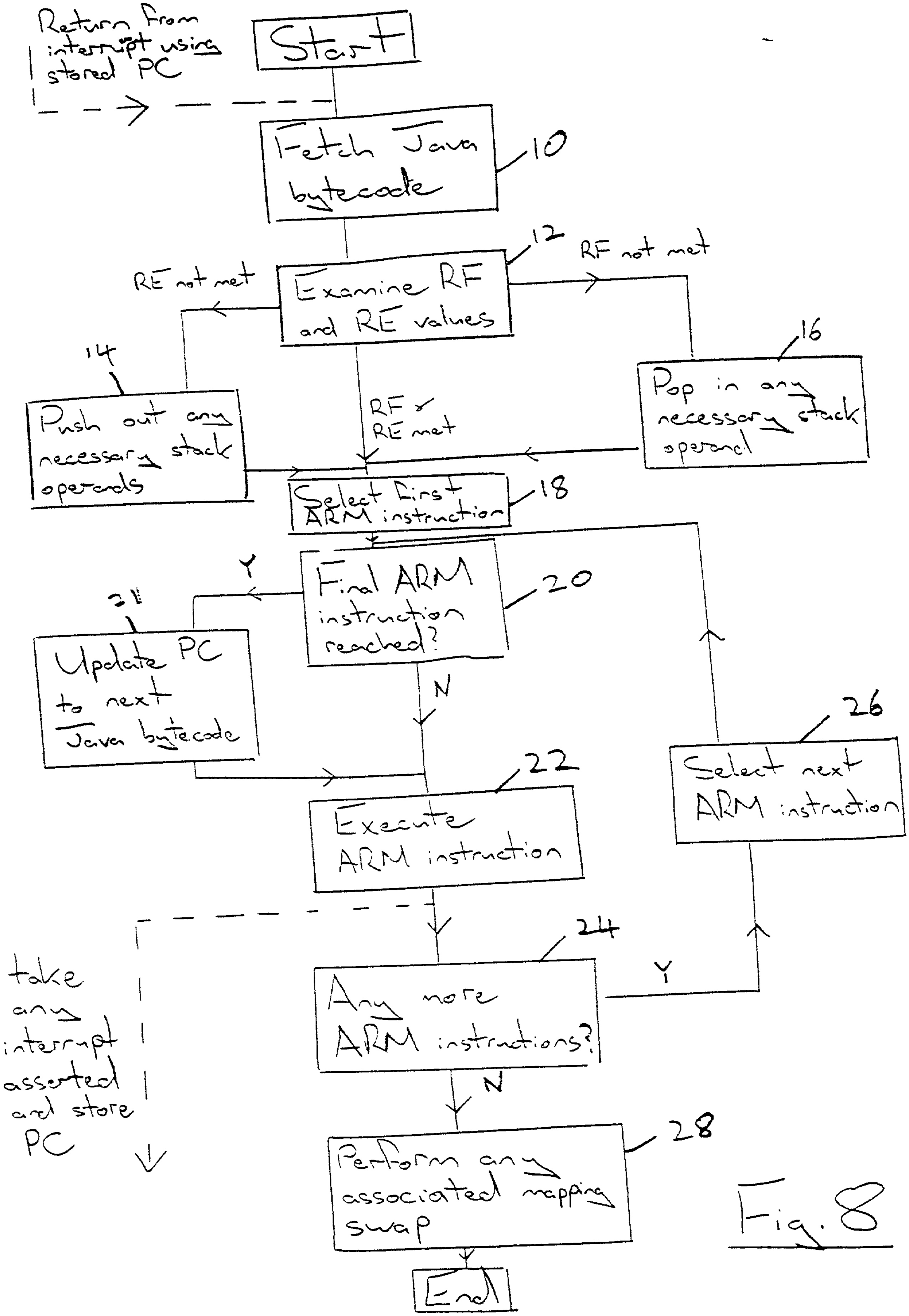
Return from interrupt using stored PC
Start
Fetch Java bytecode
10
12
RF not met
RE not met
Examine RF and RE values
14
16
Push out any necessary stack operands
RF & RE met
Pop in any necessary stack operand
Select first ARM instruction
18
Y
Final ARM instruction reached?
20
21
Update PC to next Java bytecode
N
26
22
Select next ARM instruction
Execute ARM instruction
24
take any interrupt asserted and store PC
Y
Any more ARM instructions?
N
28
Perform any associated mapping swap
End

Fig. 8

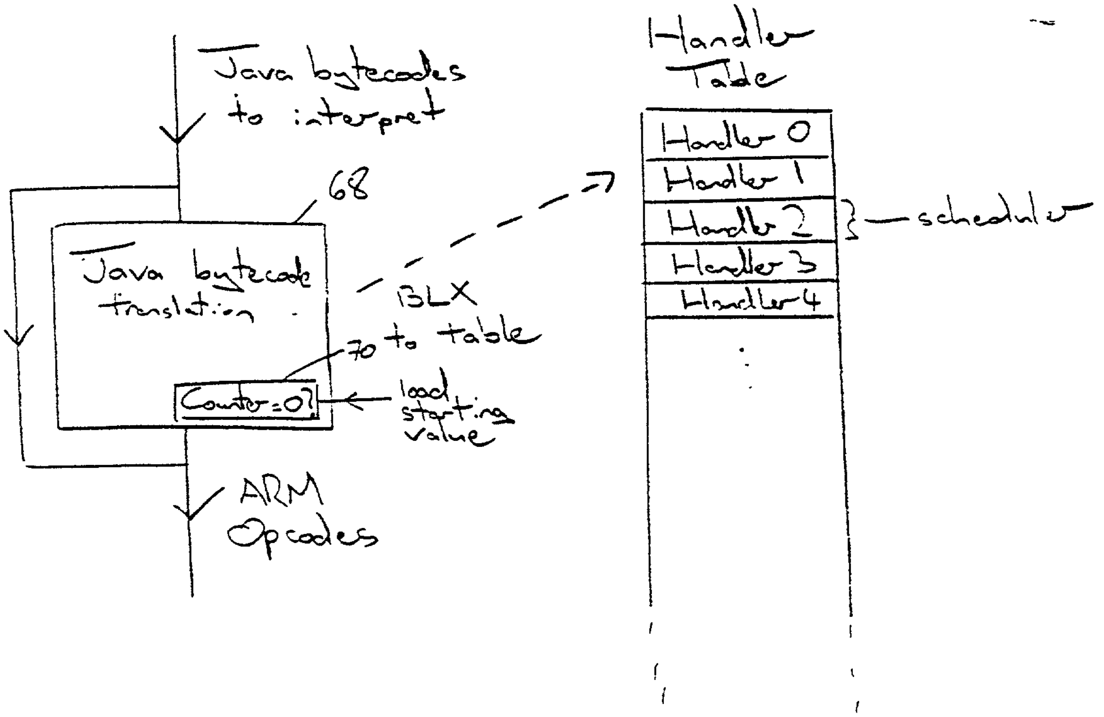
Java bytecodes to interpret
Handler Table
68
Java bytecode translation
BLX to table
70
Counter=0?
load starting value
ARM Opcodes
Handler 0
Handler 1
Handler 2
scheduler
Handler 3
Handler 4

Fig. 9

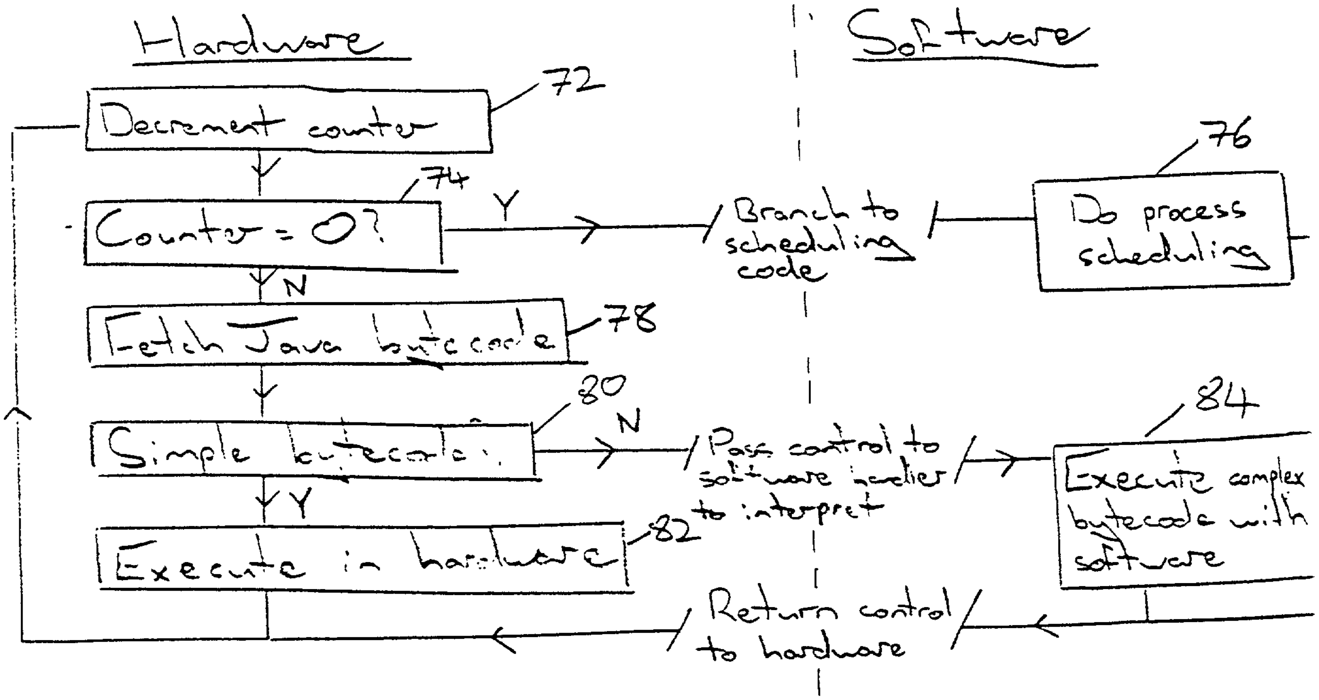
Hardware
Software
72
Decrement counter
74
Counter = 0?
Y
Branch to scheduling code
76
Do process scheduling
N
78
Fetch Java bytecode
80
Simple bytecode?
N
Pass control to software handler to interpret
84
Execute complex bytecode with software
Y
82
Execute in hardware
Return control to hardware

Fig. 10

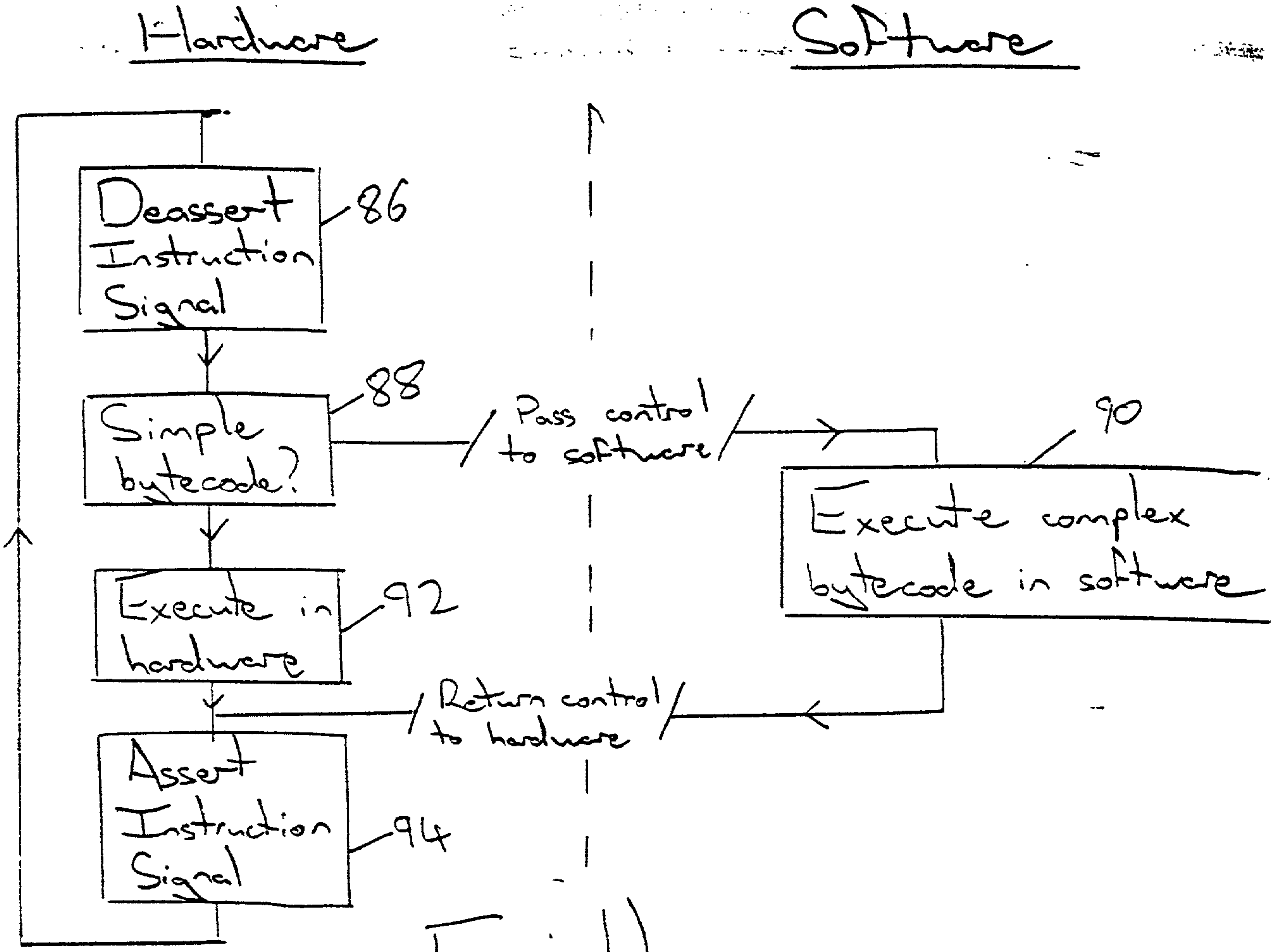
Hardware
Software
Deassert Instruction Signal
86
Simple bytecode?
88
Pass control to software
90
Execute complex bytecode in software
Execute in hardware
92
Return control to hardware
Assert Instruction Signal
94

Fig. 11

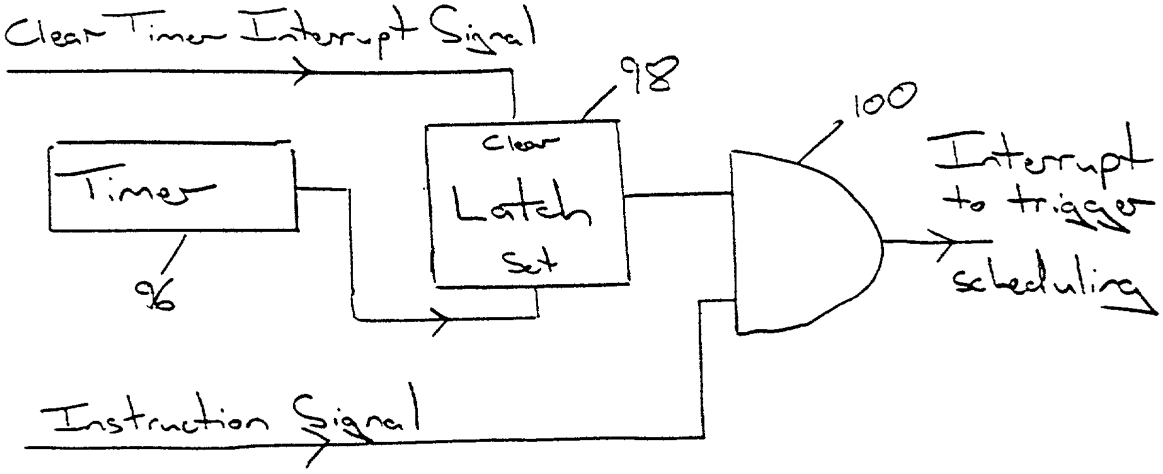
Clear Timer Interrupt Signal
98
100
Timer
96
Clear
Latch
Set
Interrupt to trigger scheduling
Instruction Signal

Fig. 12

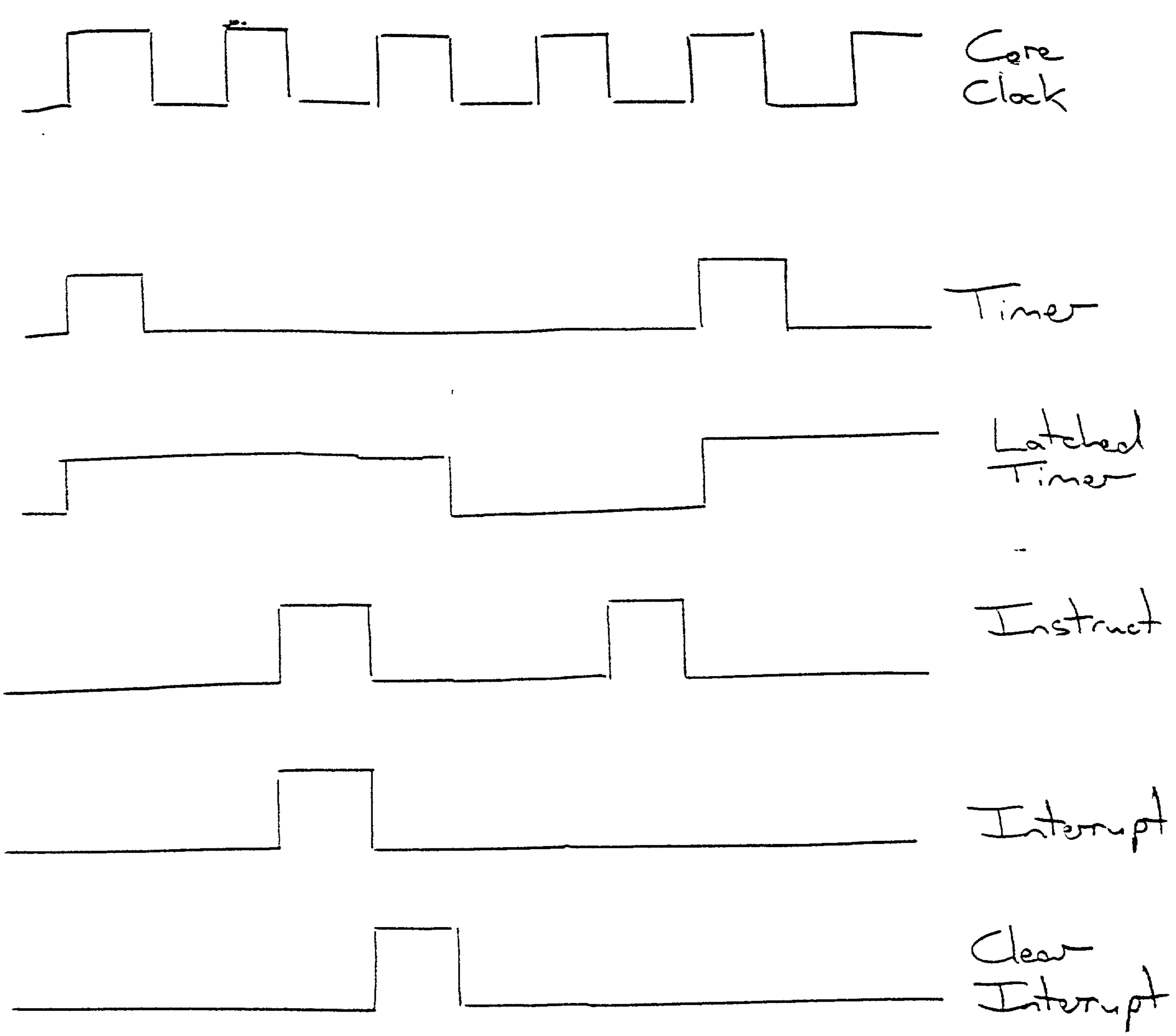
Core Clock
Timer
Latched Timer
Instruct
Interrupt
Clear Interrupt

Fig. 13